4/20/2021 *Mihaela Malita* mmalita@smith.edu

MEMO Java: https://www.java.com/

**SHELL:** https://tryjshell.org/

```
>jshell                    //to quit type /exit or CTRL+D
>javac hello.java          //produces hello.class
>java hello                //executes hello.class
// File: hello.java
public class hello {//name of class -sdame as file
        public static void main(String[] args) {
            System.out.println("hello world!");
        }// main
}//hello
//  File: Loops.java -      prints integers from 1 to limit
import java.util.*;        // for all your programs!!
public class  Loops
        public void main(String[] args) {
            Scanner cin = Scanner(System.in);
            System.out.println( "Enter a limit? "); //ask
            int limit = cin.nextInt();    //read value as int
            int i=1;  // loop counter
            while (i <= limit) {        // counts till 5
                    System.out.println(i );
                    i++;                // same as  i = i + 1;
              } //while
        } //main
        public void count(int limit){
            int i =0;
           do {   i = i + 1;            // or i++;  or  i+=1;
                    System.out.println( i) ;
            } while (i < limit) ;  //condition at the end
          for (int i = 1; i <= limit;  i++) { //same
              System.out.println(i);
          } // for
        } //count
}//class
========== Conditional =================
switch (day) {     // int day=0 to 6
        case 0:  System.out.print( Sunday");   break;
        case 1:  System.out.print("Monday"); break;
                 …
        default:  System.out.print("Error");
}//switch
Example: (n > 0) ? "positive" : "negative";
```

FORMAT: \t = tab \n = new line

System.out.printf("%.2f",3.1234567); => 3.12

Console.pause(); //waits for [ENTER]

Primitive Data Types: boolean int float double char

classes: Boolean/ Integer / Double /Character

boolean b; // true or false

IINPUT: boolean b = in. nextBoolean();

**int n = cin nextInt(); use after that**

**cin.nextLine(); // It consumes the \n character)**

d = in. nextDouble();

char ch = cin.next().charAt(0);//read a char

**Math.** Operations: + , - , /, *, % Math.PI

Math.sin(0.5) Math.cos(double)

Math.pow(3.0, 2.0) Math.sqrt(4.0)

Math.abs(-8.9) Math.log10(100) = 2

Math.random(); // [0,1) Math,round(n)

int die= (int)(Math.random() * 6) + 1 // [1– 6]

**Characters** (ASCII/UNICODE)

char ch = 'a'; //same as:

Character ch = new Character('a');

(int) ch = 97;

(char) 97 ='a'; // cast into another data type

On 1 byte: ASCII code 128: double x = Math.pow(2,7)

Read a char: char ch = in.next().charAt(0);

**boolean** : Character.isLetter(ch);

Character.isDigit(ch); Character.isWhitespace(ch);

Character.isLowerCase(ch); Character.isUpperCase(ch);

**char:** ch1 ==ch2

Character.toUpperCase (ch); Character.toLowerCase (ch);

**CONVERSIONS**

double x = 3.14; int n = (int)x; // n =3

char ch = 'a'; int n =(int)ch; // n= 97

Character.getNumericValue('5') => 5

String s = "2005"; int n = **Integer.parseInt**(s);

Binary to decimal: Integer.parseInt("100",2 )=> 4

Dec to Binary: Integer.toBinaryString(5); =>111

Dec to Hex: Integer.toHexString(16); => 10

Hex to Dec: Integer.parseInt("7591083d",16);

Array 2 String: **String s = String.valueOf(A);**

String 2 Array: **char [] A = s. toCharArray();**

String s = Character.toString(ch);

int k; String s = Integer.toString(k) ;

**String**

is **immutable** (cannot be changed!)-make new copy

String s = "Turing"; // String s = new String ("Turing");

**s.length()** **s.charAt(index)** **s.isEmpty()**

s.toUpperCase() s.toLowerCase();

s.concat("Alan") same as s = s + "Alan"; // Turing Alan

s1.compareTo(s2)// 0 if s1=s2; >0 if s1 < s2; < 0 if s1 > s2

**s1.equals(s2)** s2.equalsIgnoreCase(s1)

**s1.indexOf(s2)** **s1.contains(s2)**

**s.substring(i1,i2)** s.startsWth(s2) s.endsWith(s2)

String res = s.replace(old s1, new s2)-all s1

s.replaceFirst(s1,s2)-only the first occurrence

s.trim(); //deletes blanks before & after

s.repeat(3); //same as: s + s + s 3 times

s = "a.b.c"; //**s.split(regex) regular expression**

String [] A = s.split("."); //A = {"a", "b", "c"} s.split(".")[0]

Read: String s = in.next(); // s = in.nextLine()

**Time: import java.time.***

LocalDateTime d = LocalDateTime.now();

d.getYear(); d.getMonth(); d.getDayOfWeek();

d.getHour(); d.getMinute(); d.getSecond();

**Year.now().getValue();** // returns current Year

long t1 = **System.currentTimeMillis()**; //measure time

**Arrays**

final int n=5; // declare const size

int [] A = new int[n]; //all elements=0

int []A ={6,2,1,4,7}; //elements take values

**A.length** Arrays.sort(A); //destroys array A

Arrays.equals(A,B) // check if arrays are equal

A = B; // B is Alias Copy array use : **int B [] = A.clone();**

int [] B= Arrays.copyOf(A,many); //how # to copy from 0

System.out.print(Arrays.toString(A));

Arrays.asList(A);//convert array to list

**MATRICES**

int [][]B = {{2,4,6},{1,3,5}}; //initialize matrix 2 x 3

double [][]A = new double[n][n]; // **int n** declared before

for (int [] R : M)System.out.println(Arrays.toString( R));

**LIST** *public abstract interface extends Collection*

L.add(int i, element) L.addAll(int i, Collection col)
L.size() L.isEmpty() L.clear()
L.remove(int index) L.remove(element)
L.get(int index) L.set(int index, element)
L.indexOf(element) L.lastIndexOf(element)
L.equals(element) L.hashcode()
L.contains(element) L.containsAll(Collection col)
L.sort(Comparator comp)

---

**ArrayList:** implements the **List** interface

```
ArrayList <Integer> V= new ArrayList <>();
List <Integer> V = new ArrayList <>(5);  //size 5
Integer [] A ={2,3,4,5};
Collections.addAll(V,A);  //add array A[] into V
List <Integer> L =  new ArrayList<>(Arrays.asList(2,3,4));
L.addAll(L)
V.size();      V.set(i,w);      V.get(int i)  // same as V[i]=w
Boolean: V.add(w)   //add at the end!
V.add(int pos,w)  // inserts w in position pos
V.contains(w)      V.isEmpty();      V.clear();
V.indexOf(obj)     V.remove(int pos)
 System.out.println(V);   V1.equals(V2) ; // check if equal
T void print ( ArrayList <T> V ){ for(T x : V)
System.out.print(x);}
List <T> A1 = A.subList(0,2); //takes first 2 elements
Collections.max(V); //maximum when order is known!
Collections.sort(V);// sorting default order
Collections.shuffle(V);
Comparing objects (Example Person)
In main():  Collections.sort(A,Person.idComp);
add in Person class:
public static Comparator <Person> idComp = new
Comparator<Person>() {
public static int compare (Person s1, Person s2) {
        return s1.id - s2.id;   }};     // int id
public static Comparator <Person> nameComp = new
Comparator<Person>() {//if compare by name(String)
public int compare (Person s1, Person s2) {
        return s1.name.compareTo(s2.name);   }};
```

---

**OOP**

```
class Person { //javac Person.java => Person.class
        private String name;  private int yob;
        public Person(String s, int y){
                setName(s); setYob(y);}
public void setName(String name){ this.name = name;}
    public void setYob(int yob){ this.yob = yob;}
    public String getName(){ return name;
      public int getYob(){ return yob;}
     public String toString() { return (name+"\t"+ yob);}
}//class
public class Program {  // file:needs Person.class
public static void main(String args[]) {
        Person P = new Person("Ada",2000);
        System.out.print(P+ " YOB:"+ P.getYob());
}}// driver class
```

---

**IO – Files** **import java.io.*;**

```
String s = System.Console().readLine(();
```

**WRITE in file:**

```
PrintWriter fout = null;
try { fout = new PrintWriter(new File(filename));}
catch (IOException ex ) { System.out.print(ex); }
fout.write("Hello world!");
 fout.close();
```

**READ from file:**

```
Scanner fin = null;
try  {   fin = new Scanner(new File("myfile.txt")); }
catch ( IOException ex) { System.out.print(ex); }
while ( fin.hasNext() )  {  // checks  end of file
        String s = fin.nextLine();
        System.out.println(s);
}  fin.close();
```

---

**LinkedList- is a List but has more methods!**

```
LinkedList <Integer> LL = new LinkedList <> ();
var LL =  new LinkedList<>(Arrays.asList(2,3,4));
LL ==> [2, 3, 4]
```

part of the Java Framework Collection framework (JFC)
Same method inherited from **List** as **ArrayLis**t:
LL.set(int i, w)  LL.add(w) same as addLast()
LL.add(int i, w)- inserts index i
LL.clear()  LL.size()  LL.set(int i, w)
LL get(int i)  LL.indexOf(T w)  LL.contains( w)
LL.addAll(List)  LL.remove(int index)
LL.removeFirstOccurence(w)
*More methods:*  LL.pop()  - remove first
LL.peek()- display first;  LL.push(T w) – add first
LL.addFirst(T w)  LL.addLast(w)
LL.getFirst()  L.getLast()  LL.lastIndexOf(T w)

```
LL.sublist(i1,i2); // returns sublist from indexi1 to i2
LinkedList L2 = new LinkedList <Integer>(L1); //copy
```

Integer A[] = new Integer [LL.size()];//declare A
Copy LL to A[]:  **LL.toArray(A);**
ListIterator<E> listIterator(int index)

---

**Stack LIFO**

```
Stack <T> S = new <>Stack();
```

S.isEmpty()  S.size()  S.peek()  //shows top
S.pop()  //removes top  S.push( e) //adds  S.search(e)

---

**Queue FIFO – "Line at a Cefeteria"**

public interface **Queue<E>** extends Collection<E>

```
Queue<String> Q= new LinkedList<>();
```

Q.isEmpty()  Q.size()  Q.add(w)
Q.peek() first e  Q.remove()  (first)  Q.poll()

**PriorityQueue Class**

```
PriorityQueue <T> Q =  new PriorityQueue <>(new
myComp());
```

void clear() int size() add(E e); contains(w); peek()  poll()
boolean remove(w);  myArr[].toArray(Q)
class **myComp** implements Comparator <T>{
public int compare(T x, T y){ return x.length()- y.length();
}}

---

**HashMap/dictionary**

```
HashMap <T,T> H = new HashMap<T,T>();
```

H.clear()  H.size()  H.isEmpty()
H.put(k,value)  H.get(key)  H.keySet()  H.values()
H.containsKey(k)  H.containsValue(v)
H.remove(k,v)  H.replace(k,oldv,newv)

---

**Set**

```
Set <T> s1 = new HashSet<>(Arrays.asList(myArray));
 Set<T> s2 = new HashSet<>(s1);
 Collection<T> noDups = new HashSet<>(c);
```

s1.containsAll(s2) // s2 is a **subset** of s1
s1.retainAll(s2)  **//intersection**
s1.addAll(s2) // **union**
s1.removeAll(s2) // s1-s2 **difference**